Milwaukee Area Atari Users Group

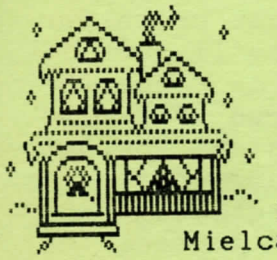
MILATARI NEWSLETTER

VOL. 5 NBR. 1 PRICE $1.50 DECEMBER 1985

DECEMBER 21,1985
ARMBRUSTER SCHOOL - GLENDALE

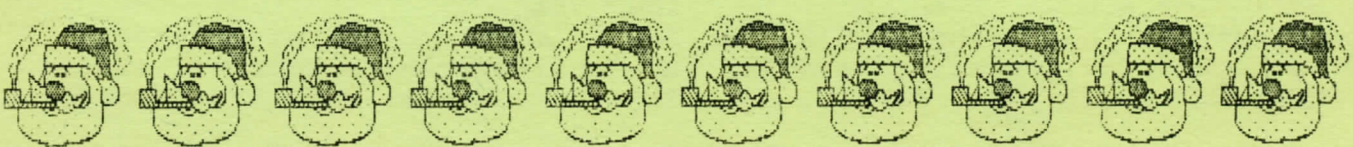
Adgenda

2:15 PM - 520ST SIG Meeting
2:15 PM - C Language Class
2:15 PM - 256K Upgrade to 800XL Workshop
3:30 PM - 2nd ANNUAL HOLIDAY CONTEST Presentation & Judging by the membership.



SPECIAL ANNOUNCEMENT

Please register your HOLIDAY CONTEST entry with Carl Mielcarek 355-3539 ASAP. So far there are no contestants (easy competition). Door prizes will be awarded if contest flops. Carl also asks that you bring a small treat to share with the membership - cookies, candy etc. if possible. Coffee, Hot chocolate and egg nog will be served.



THE FUZZY NOLAN REVIEW

BY GARY NOLAN

WE WISH YOU A MERRY CHRISTMAS
WE WISH YOU A MERRY CHRISTMAS
WE WISH YOU A MERRY CHRISTMAS
....AND-A-HAP-PY, NEW-YEAR!!!

THINGS ARE LOOKIN' UP

Every year InfoWorld, a weekly newspaper devoted to the microcomputer end of the business, selects products of the year in several catagories. This years winner of the Hardware Value of the Year was none other than Atari's 520ST. Not an Oscar or Emmy but reconition none the less. The 520 was also the hit of the PC World Show held in jolly old England earlier this year. With the release of more and more software, '86 is shaping up as a very good year for the new systems.

Add the Computer Curriculum Corp. to the list of ST believers. CCC plans to use the 520's as a workstation in its CCC Learning Station system. Each station will consist of a 520, two floppy drives and a hard disk. It will be connected to Microhost superminicomputer via a modem or a 6-wire cable. The Microhost is a 68010 based computer with a 224Meg hard disk that can accomodate up to 128 simultaneous users. Each CCC 520 will come with cost between $2-3000 and come with Basic, Logo, Neochrome and ST Write. CCC has added color to the Microhost educational software for use with the ST's. In all CCC has 2500 hours of courseware running under Unix System V, and claims that more than 1300 schools and half a million students use the Microhost System. Looks like another step forward.

COME ONE, COME ALL

The first meeting of the ST SIG was held at the Nov. meeting and was attended by about 10/12 people, not all of them ST owners. The SIG is open to those who own or are interested in the ST line of computers. For now the SIG meetings will be held before the regular meetings. Milatari plans to build and support a public domain library for the ST's in the same way it has for the 8-bit systems. As other models are added to the ST line, they will receive the same support. So if you own, plan to buy or just want to learn more about the new computers, stop by the next SIG meeting and check it out.

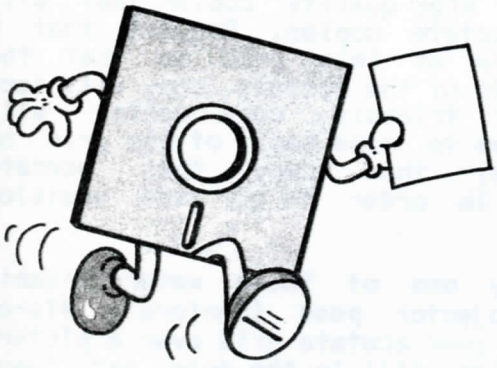
SAY BYE.....

One more computer mag bites the dust. This time it's one of the oldest. Creative Computing announced that the December issue would be its last. The magazine hade been published for 11 years and was the first one I ever bought or subscribed to, before I had a computer even. It had gotten thinner in the past year but what appeared was still pretty good.

LISTEN UP

All you Prometheus modem owners might want to pay attention to the following. If you already have the Options Processor board installed you can upgrade to the 512K RAM Communication Buffer Option for only $40. This does not include the chips of course, but the lowest price for 256K-150ns chips is $2. You'll need 16 of the little beauties so the whole-hog update will run $72 plus shipping. Among other things you get with it is a serial printer port. It also gives you password protection and auotmatic call back to protect the files you have stored in that 512K of memory. You could almost almost set up a BBS with the thing. If you're interested come to the next meeting and we will be able to give you more information on all the features of this upgrade.

JAMCO Electronics of Belemont, CA is the only current source for the board. Their number is (415) 592-8097.



FROM THE DISK OF

Dave Frazer

NOVEMBER MEETING

Another exciting meeting is being planned for December. We lead off with a workshop and 'C' class at 2:15pm. The 520ST SIG will also meet at 2:15pm. See my notes below for the activities of this SIG. This months workshop will be an actual demonstration of the 256K upgrade done on a 800XL. And Mark Manyen continues the 'C' class at the same time.

At 3:30, we will hold our 2nd annual Christmas programming contest. Members in several categories on competition will demonstrate their programs of a Christmas theme. The membership will vote for their favorites and prizes will be given. It's not to late to enter. Give Carl a call to reserve presentation time. Cookies, egg nog, hot chocolate and coffee will be served. Watch for a jolly visitor.

Libraries will be open at 2:00pm. They will be closed during the Christmas program presentations.

Mark your calendar for Saturday, December 21st. The location is Ambruster school. The fun begins at 2:00pm.

WE NEED A SECRETARY

We regret to announce that Jim Mangione has submitted his resignation as secretary of MILATARI. We will miss Jim's leadership on the board and will hold him to his promise to help the club as his time permits.

I would like someone to come forth and volunteer to serve as Secretary. Really folks, it's not that hard, and it's also interesting work. Here's a chance to give something back to the group. How about it -- any volunteers?

Call me or see me at the next meeting if you are interested.

520ST SIG

A small band of pioneers led by Gary Nolan, met at our November meeting to put together the beginnings of the 520ST special interest group (SIG). This SIG will continue to meeting on MILATARI Saturdays to foster and promote the interest in Atari's 520ST computer. The group has started a disk library. ST-BASIC is available to our members who have not yet received it from Atari. Copies of the ST-BASIC manual can be ordered from SIG member Roy Duvall.

The SIG meets again on December 21st at 2:15pm. If you have a 520ST or are planning to get one or are just curious, please join in.

RENEW MEMBERSHIPS

Many of us will find membership renewals in the mail in the coming months. To continue the activities of your group and to insure our leadership in the Milwaukee Atari community, please remember to get your check to Steve Tupper. We need your continued support!

GENIE SERVICE

General Electric Information Services, the world's largest commercial teleprocessing network, is offering SIGs, software, CB, e-mail & games for personal computers. Non-prime time rate (6pm-8am, Sat, Sun & holidays) 300 or 1200 baud for $5.00/hr. Sign up from your keyboard.

1-800-638-8369 Half duplex, 300 or 1200 baud. Upon connection enter 'HHH' then press return. At the U#= prompt enter 'VJM11951,GENIE' and press return.

Try is at that number for 5 free minutes before you sign up.

Printer

GRID BY KEN WATSON

PRINT SHOP DESIGN GRID

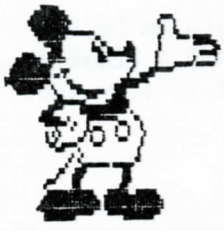
Below you will find the most helpful tool available for designing your own Print Shop Graphics. Ken Watson, Elsa, Yukon, Canada, sent us this excellent grid, and we will give you some tips on how to make the most of it.

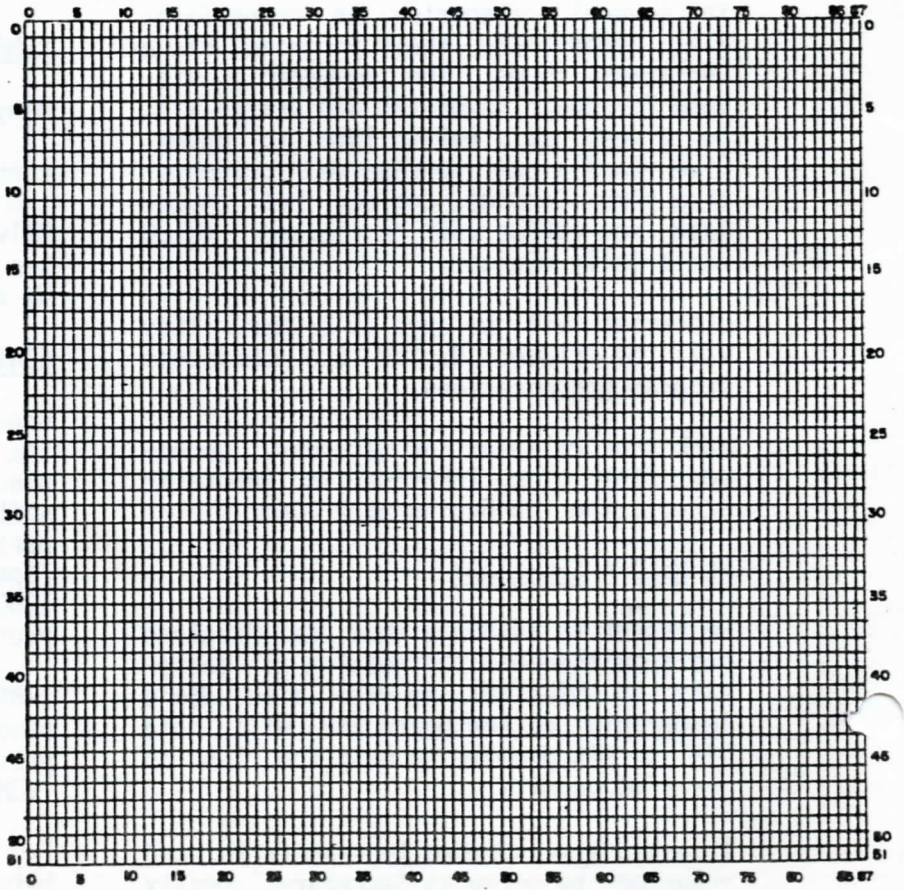
The most obvious useage of the grid is to physically design your Print Shop graphic on photocopy of this grid. If you are not necessarily artistic in nature (like most of us), here is another way you can make use of this grid.

Find a high quality copier that will give you acetate copies. Request that a copy be made of this grid so that the emulsion side of the acetate copy will face down. Your friendly copy center will probably have to make copy of the grid on one acetate, then copy that acetate upside-down in order to get the emulsion down.

Now buy one of those water-soluable overhead projector pens (Sanford's Vis-a-Vis), place your acetate grid over a picture of your choice, fill in the dots, and input it into the graphics editor of the Print Shop Master Diskette. It is as easy as that to produce custom graphics like those you see below.

PRINT SHOP Design Grid





KRYING THE ATARI
By Gerald Hagopian

The joystick jacks on the Atari are unique. They are actualy input/output devices of some sophistication. We will only be making use of a small portion of their abilities to make a real key. That's right, a physical key that plugs into the computer.

THE ATARI KEY

In case you didn't know, I'll quickly review the way the Atari paddles work. A potentiometer in the paddle acts a variable resistor to the current supplied by the computer. The amount of resistance is translated by the computer into a numeric value from 0 to 255. By using some inexpensive parts, a soldering iron, and some patience, we are going input a given resistance into the Atari. This will give us a numeric value that will in effect be a code.

First the parts. Hop down to your local Radio Shack or electronics parts store and pick up the following.
(1) 9 pin female DIN plug
part# 276-1538 $2.49
(1) hood for the above
part# 276-1538 $1.99
(2) packs of 1/4 watt resistors
each between 4.7K ohms and 470k ohms
(2) 2x56 machine screws and nuts

You'll also need a soldering iron, some rosin core solder, electrical or masking tape, a sharp craft knife, and a small screw driver to put everything together.

Now look at Figure #1. You're seeing the back (the part that goes inside the hood) of the plug. Very carefully clip the wire leads of the resistors, something like the way they are in Figure 2. We want to connect pin 5 and 7, and pin 9 and 7 with those leads. Carefully solder them in place and put a small piece of tape between them so they remain separate. That's it! Close up the housing and you're done. You may have to trim the front plastic tabs a bit so the 'key' inserts smoothly into port one of you Atari. Use the two small machine screws and nuts to secure the plug to the hood, through the two holes in the front of the assembly.

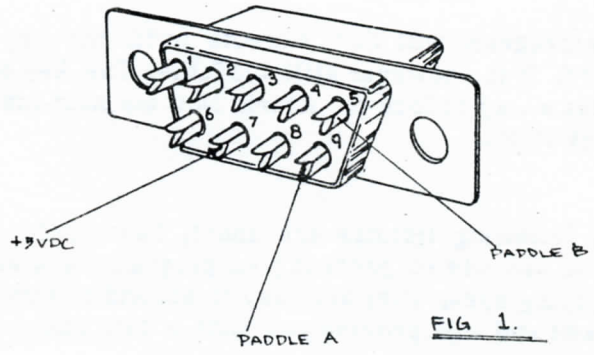


FIG 1.

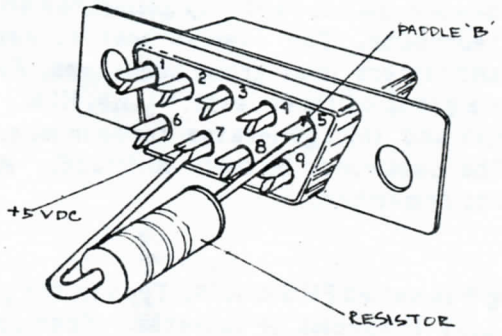


FIG 2.

Now BOOT your Atari with the basic cartridge installed. Type in the following lines...
```
10 A=PEEK(624):B=PEEK(625)
20 PRINT "VALUE A=";A;:PRINT "  VALUE B=";B
30 GOTO 10
```
Plug in the 'key' to joystick jack 1 and run the program. The values you see are the values of the locations 624 (decimal) and 625 (decimal). Write them down, these are your code numbers for your key.

USING THE KEY

LOAD a basic program into memory then list it. At the beginning of the program (I usually use lines 2,3 and 4) insert the following lines.
```
2 IF PEEK(624)= [YOUR "A" VALUE] AND PEEK(625)=
[YOUR "B" VALUE] THEN GOTO 4
3 PRINT "INSERT KEY TO RUN":GOTO 2
4 REM.
```

MORE ON PAGE 6

The program will not execute until the key is inserted. It is, however still LISTable. The key only provides a way to lock our Atari. Now we must install the lock itself.

The following listings are short, basic programs that you can use to protect your programs and disks from prying eyes. They are easy to append to existing programs and will provide you with a fair amount of protection.

Listing 1 is called PASSGEN. It will provide you with a password that is randomly generated and fairly hard to reproduce. The easiest way to remember numbers and letters is in group sequences. PASSGEN generates a group of three letters (i.e. KLM) inserts and asterisk and then generates three numbers (i.e. 396). The password is then KLM*396. An easy sequence to remember.

Listing 2 is called PRGMPASS. Type in the program and then LIST it to disk or cassette. When you want to password protect a program, ENTER it into your listing. Type in your password on line 30000 and the program name at line 32100 (D:FILENAME). If your are using your key insert your code numbers on line 30000To protect your program, type GOTO 32000.

Listing 3 is called PASSPRO1. It is almost identical to PRGMPASS, but is made to be set up as a disk protector. Type the listing in. Insert your password and code numbers, and save it to disk by typing GOTO 30000. Now use your autoboot generator to create an autorun file from PASSPRO.

When you boot your disk the program will not let you access files until you enter the correct password. Failure after three tries will put the Atari into memo pad mode.

A word of warning on this listing and the preceding. Once you have protected the program, it will not be able to be listed or loaded. System reset will re-boot the disk. The break key will break the program but will not allow a listing of it. Your program can still be copied, but unless the copier knows the password and the key, the program will not run.

IT IS VERY IMPORTANT THAT YOU SAVE AN UNPROTECTED BACK-UP COPY OF YOUR PROGRAM ON A FILE DISK.THE PROGRAM WILL ONLY ALLOW YOU TO RUN YOUR FILE, NOT LOAD OR LIST IT.

ADDITIONAL CODING

By using jumper wires we can put even more codes into the key. Location 632 is the STICK0 register. It returns a number representing a direction when you move your joystick. Their are nine possible numbers to choose from. The following list gives the pin numbers (see figure 1) and the number returned at location 632 when they are connected to ground. If no connections are made then PEEKing (632) will return a value of 15. Use insulated wire when making the connections. Location 644 is STICKTRIG0. Its normal (unconnected) state is 1. Connecting it will return a 0.

If you use all of the options we've talked about, you'll end up with four different numbers encoded your key.

Now you have a key that, along with the protection programs, will allow you to secure those valuable programs from prying eyes.

LOCATION 632 (JOYSTICK 0)
GROUND PIN #8

PIN	VALUE	PINS	VALUE
1	14	1 & 4	5
2	13	4 & 2	6
3	11	2 & 3	9
4	7	3 & 1	10

NO CONNECTIONS VALUE=15

```
 1 Rem PASSGEN
 10 Dim Pas$(9)
 20   For N=1 To 3
 30   Pas$(N,N)=Chr$(Int(Rnd(0)*90))
 40   If Pas$(N,N)<Chr$(65) Then 30
 45   Next N
 46 Pas$(4,4)="*"
 50   For N=5 To 7
 60   Pas$(N,N)=Chr$(Int(Rnd(0)*57))
 70   If Pas$(N,N)<Chr$(48) Then 60
 80   Next N
 90 ? "PASSWORD GENERATED IS":? Pas$
```

Mother Hubbard's Cupboard or the Treasurer's Report

BALANCE	10/31	416.94
	INCOME	
Membership		140.00
Cash on hand		50.50
Sales		456.00
	EXPENCES	
Disks		250.00
Change		50.00
BBS		53.46
Nov. N.L.		261.45
Rent		67.50
Postage		10.89
BALANCE	11/30	340.14

ATR8000 and CP/M SIG SCHEDULE

The SIG meets in the home of Joe Kasper at 8PM on the 2nd Thrusday of the month. Please call Joe for the adjenda at 782-9041.

January	9,1986 at 8:00PM
Feburary	14,1986 at 8:00PM
March	14,1986 at 8:00PM
April	11,1986 at 8:00PM
May	9,1986 at 8:00PM
June	13,1986 at 8:00PM
July	No meeting
August	No meeting
September	12,1986 at 8:00PM
October	10,1986 at 8:00PM
November	14,1986 at 8:00PM
December	No Meeting

Keying The Atari

From Page 6

```
0 Rem PRGMPASS
1 Poke 580,1
2 Dim Try$(10),Pas$(10):Err=Peek(195):At
mpt=1:Poke 580,1
3 If Peek(624)=43 And Peek(625)=23 Then
Gosub 30000
4 Position 15,10:? "INSERT KEY":Goto 3
30000 Pas$="YOUR WORD":? " "
30010 Position 1,8:? "INPUT PASSWORD":Po
sition 1,10:Input Try$:If Try$=Pas$ Then
 Goto 30050
30020 Atmpt=Atmpt+1:? "PASSWORD INCORREC
T.  TRY AGAIN":If Atmpt=4 Then 30040
30030 Goto 30010
30040 ? "PASSWORD INVALID....PROGRAM ABO
RTED":  For Pause=1 To 500:Next Pause:By
e
30050 ? "PASSWORD VALIDATED":  For N=1 T
o 300:Next N:Return
32000   For Vari=Peek(130)+Peek(131)*256
 To Peek(132)+Peek(133)*256:Poke Vari,15
5:Next Vari
32100 Poke Peek(138)+Peek(139)*256+2,0:S
```

```
1 Rem PASSPOR1
2 Dim Try$(10),Pas$(10),File$(15),D$(19
:Err=Peek(195):Ak=43:Bk=23
3 If Peek(624)=Ak And Peek(625)=Bk Then
30000
4 ? "INSERT KEY":Goto 3
30000 Pas$="G HAGOPIAN":Atmpt=1:Poke 58
,1:? " "
30010 Position 1,8:? "INPUT PASSWORD":P
sition 1,10:Input Try$:If Try$=Pas$ The
 Goto 30050
30020 Atmpt=Atmpt+1:? "PASSWORD INCORRE
T.  TRY AGAIN":If Atmpt=4 Then 30040
30030 Goto 30010
30040 ? "PASSWORD INVALID....PROGRAM AB
RTED":  For Pause=1 To 500:Next Pause:B
e
30050 ? " ":Position 5,5:? "FILE LISTIN
":Open #1,6,0,"D:*.*":Trap 30070
30060 Input #1;File$:? File$:Goto 30060
30070 If Err Then Close #1:Trap 40000
30075 Trap 30110
30080 ? :? :? "ENTER FILENAME TO RUN"
30090 File$="":Input File$:D$="D:":D$(L
n(D$)+1)=File$
30100 Run D$
30110 If Err Then ? "ERROR # ";Err:Trap
40000:Goto 30080
32000   For Vari=Peek(130)+Peek(131)*25
 To Peek(132)+Peek(133)*256:Poke Vari,1
5:Next Vari
32100 Poke Peek(138)+Peek(139)*256+2,0:
```

CHRIS CRAWFORD ASSEMBLY LANGUAGE COURSE

LESSON FIVE: INDEX REGISTERS & LOOPING

We are now going to expand the model of the 6502 that you have been using. Until now, the 6502 I have described had nothing more than a status register, program counter, and accumulator. Now I am going to reveal the existence of two new registers in the 6502: the X- and Y-registers.

These two registers are eight-bit registers just like the accumulator. You can load numbers into them and store them out just as you can with the accumulator. You cannot do arithmetic or Boolean operations with them as you can with the accumulator. But you can do a number of very special things that greatly increase the power of the 6502.

Let's start with the simple move instructions. The first are LDX and LDY, which load the X- and Y-registers the same way that LDA loads the accumulator. Then there are STX and STY, which store the X- and Y-registers the same way that STA stores the accumulator. There are also four commands for transferring bytes between registers; these are TAX (transfer A to X), TAY (transfer A to Y), TXA (transfer X to A), and TYA (transfer Y to A).

Then there are four special instructions that you will use very often. These are INX and INY, which increment (add one to) the X- and Y-registers, and DEX and DEY, which decrement (subtract one from) the X-and Y-registers.

Finally, we have the CPX and CPY commands, which compare X or Y with the operand of the instruction. These two instructions operate in exactly the same way that the CMP instruction works, except that they use the X- and Y-registers instead of the accumulator.

What are these two registers used for? Well, they are sometimes used as temporary registers. If you are in the middle of a lengthy computation, and you need to save a value currently in the accumulator to make room for something else, the X- and Y-registers are a handy place to stuff values away for temporary storage. Programmers do this all the time.

However, temporary storage is not the real purpose and vaue arise from their utility as index registers. Index registers go hand in hand with loops; the best way to show you how they are used is to dump the whole schmeer at once and then explain it.

CHRIS CRAWFORD ASSEMBLY LANGUAGE COURSE

So consider the following problem: your program has to deal with the possibility of user errors. Suppose you require the user ty type in a file name for your program to read. What happens if this file is not on the disk? You have to put an error message on the screen that says, "FILE NOT ON DISK!" How do you print the message? Here's a sample bit of code that will do it:

```
LDX    #(ENDMSG-ERRMSG-1)
LOOP1 LDA    ERRMSG,X
STA    SCREEN,X
SEC
SBC    #$20
DEX
BPL    LOOP1
JMP    ELSWHR
ERRMSG DB    'FILE NOT ON DISK1'
ENDMSG DS    1
```

Let's take apart this code and explain it step by step. First thing we do is load the X-register with the number of characters (minutes one) in the message. The expression (ENDMSG-ERRMSG-1) will calculate that length at assembly time. This turns out to be 17 characters long. If we were pedestrian about it we could have just written LDX #16, but this way, if we decide to change the message we don't have to remember to go back and change the LDX command. Neat, huh?

OK, so now we have a 16 in the X-register. Now the 6502 comes to the next command -- LDA, ERRMSG,X. This command tells it to load the accumulator with the byte at (address ERRMSG, indexed by X). What this means is as follows: the 6502 will take the address ERRMSG and add the value of the x-register to that address. It will then go to the address so calculated and load the accumulator with the contents of that address. Since X contains a 16, the 6502 will go to the 16th byte after the first byte in the table ERRMSG. If you count characters, you will see that the 16th byte is the exclamation point. So the 6502 will load the ASCII code for an exclamation point into the accumulator.

The next two instructions (SEC, SBC #$20) are necessary to correct for the Atari's nonstandard handling of ASCII codes. They make sure that the exclamation will be printed on the screen as an exclamation point.

The next instruction (STA SCREEN,X) stores the reslt indexed by X. The 6502 will add the contents of X (still 16) to the address SCREEN. It will then store the contents of the accumulator into that address. If that address is part of screen RAM, then you will see an exclamation point appear on the screen.

The next instruction that the 6502 encounters is the DEX instruction. This instruction subtracts one from the X-register,

CHRIS CRAWFORD ASSEMBLY LANGUAGE COURSE

Next, the 6502 comes to the instruction BPL, LOOP 1. This will branch if the N-flag is clear. The vaue of the N-flag is affected by a DEX instruction. The value of bit D7 of the result is transferred to the N-flag. Bit Dd7 of 15 is a zero, hence the N-flag is clear, hence the 6502 will indeed take the branch. Note that it branches back up to LOOP 1.

Now it will repeat the process, only this time X contains a 15, not a 16. It will therefore grab the 15th character, an ASCII 'K', and store that to the screen position just before the exclamation point. Then it will subtract one form X to get a 14, and will continue the loop.

This process will continue, with the 6502 grabbing bytes in reverse order form the table and storing them onto the screen, until after the 6502 does the seroth byte. When X contains a zero, and the 6502 executes a DEX, it obtains the result $FF. This sets the N-flag. When the 6502 encounters the BPL command, it will NOT take the branch; instead, it will skip the branch and go on to the JMP statement. The loop is terminated.

In this one fragment of code you have seen two major ideas: indexed addressing and looping. They are so closely related that it is hard to talk about one without talking about the other.

You can use indexed addressing with either the X-register or the Y-register. You most commonly use indexed addressing with the LDA and STA commands, but you can also use it with many of the other 6502 commands: ADC, SBC, CMP, AND, ORA, EOR, LSR, ROR, ADL, and ROL can all be used with indexed addressing. Indexed addressing allows you to work with tables or arrays of data.

There is one ugly catch: all of your arrays must be less than 257 bytes long, because the index registers are only eight bits wide. Most of the time this is not a serious problem. However, if you must address a larger table or array, you can use indirect addressing. To do this, you calculate the address that you desire to access, store that address in two contiguous bytes on page zero (low, then high) -- we call these two bytes a pointer -- and then refer to the pointer like so:

```
LDA   (POINTER), Y
```

This instruction will take the address out of pointer, add the value of Y to it, and load the accumulator with the contents of the address so calculated. If POINTER contains $4567 and Y contains a 2, then the 6502 will load the acumulator with the contents of address $4569. You are still restricted by the size of Y, but you can always go back and change the POINTER if you need to span larger arrays. In this case, you frequently just leave Y equal to zero and do all of your indexing directly with changes to POINTER.

CHRIS CRAWFORD ASSEMBLY LANGUAGE COURSE

The last topic I will take up is termination techniques. Every loop must somehow be terminated if you are to avoid the problem of the Sorcerer's Apprentice. You will note that the programming example I gave used a rather odd approach. I started at the end of the array and worked backwards. Why not start at the beginning and work forwards? It's slightly more efficient going backwards than forwards. When you go forwards, you have to terminate the loop with:

```
INX
CPX   #17
BNE   LOOP1
```

Whereas when you go backwards, you need only use:

```
DEX
BPL   LOOP1
```

Going backwards you save one instruction. However, if this confuses you, feel free to count forward; that works, too, only it's a little less efficient.

There is also a problem on choosing whether to BNE or BPL. BPL restricts you to a range of only 127 bytes, but BNE, but index from ERRMSG-1 and SCREEN-1 instead of ERRMSG and SCREEN.

There are lots of other sneaky ways to terminate loops, but they fall into advanced topics.

Notes on Quarter Meg

In September 1985 vol.10 no.9 Byte the small systems journal there was an article called The Quarter Meg Atari 800XL By Claus Buchholz. In this article Claus explains how the ram works in the Atari computer and most important how to perform the upgrade to 256K byte on an 800XL.

There are some important points to remember about this upgrade. First upon power up the computer looks like a standard 800XL with no extra ram. Second there are eight 32k banks and switching can be the upper 32k or lower 32k of ram. Third there are two types of 800XL's with different video controllers and care is in order to check which circuit to use for your machine. Finally the software for this ramdisk will not work on the 130XE.

I also do not recommend the novice to perform this upgrade with out help from a more experience tech. The ram chips are sensitive to static electricity, and there are wiring changes to the computer. But in all this upgrade may suite your needs with a ram disk that is equal to a full double density single sided disk drive that works at the speed of the computer. That costs about 1/6 less,than said disk drive and power supply.

PARTS LIST

8 41256 256k-bit dynamic RAM (200ns or faster)
1 ----- 33-ohm, 1/4 watt resistor
1 276-150 Radio Shack 2- by 3-inch circuit board
2 ----- 16 pin IC socket
1 ----- 18 inch 25 conductor ribbon cable (for video controller CO21697 or circuit fig.1)
1 74LS153 Dual 4-to-1 multiplexer (for video controller CO12296 or circuit fig.2)
1 74LS151 8-to-1 multiplexer
1 74LS393 Dual 4-bit counter
1 74LS00 Quad NAND gate

I could not get a 1/4 watt resistor so I used a 1 watt resistor instead.

(note the pin numbering of the chips)

```
          notch
Pin 1 => !-----U-----! <= Pin 16
Pin 2 => !   CHIPs   ! <= Pin 15
Pin 3 => !           ! <= Pin 14
```

It took about 2 hours to make the circuit board and about 1 hour to install the circuit board in the 800XL computer. But the hardest part was getting the a good copy of the software. Since BYTEnet is a password BBS and will not let you in at all with out a password, I had to resort to other means to get the software to work the ramdisk. This software is on the MILATARI BBS 414-781-5710 for down loading.

This ramdisk works with Dos 2.0, OS/A+, and Mydos. The ramdisk will work if you have Omnimon but the ramdisk handler will over write Ominmon. I under stand here is a basic for this 256K upgrade and as soon as I get a copy to look at I will write a review of it. If you are interested read the for mentioned BYTE for the full information on the upgrade Have fun.

By Peter Kurth

Reprinted from the
I/O Connecter

```
5 REM BASIC DENSITY CHANGING PGM BY
 R. HIATT 1/5/85
6 REM Called from Basic by USR rout
ine, ie. X=USR(ADR(I$),#,D) where #
 is drive No. and D is "D" or "S" i
n Dec.
10 DIM I$(131)
20 FOR I=1 TO 131:READ A:I$(I,I)=CH
R$(A):NEXT I
100 DATA 104,104,104,201,1,144,26,2
01,3,176,22,24,141,1,3,169,49,141,0
,3,104,162,0,104,201
110 DATA 83,240,6,232,201,68,240,1,
96,134,218,169,64,141,3,3,169,78,14
1,2,3,169,12,141,8
120 DATA 3,169,0,141,9,3,169,203,14
1,4,3,169,0,141,5,3,32,89,228,48,21
8,165,218,208,4
130 DATA 162,0,240,2,162,4,134,208,
201,1,240,6,162,128,160,0,240,4,162
,0,160,1,132,209,134
140 DATA 210,201,2,240,4,162,18,208
,2,162,26,134,206,169,128,141,3,3,1
69,79,141,2,3,32,89
150 DATA 228,32,224,7,96,0
```

BLOCK I/O WITH ATARI BASIC

David Pelinka

The ability to read or write a block of characters from a data storage device with BGET and BPUT is one of the many nice features of BASIC XL. It is much faster than handling one character at a time and you can read or write an entire file as fast as the DOS LOAD and SAVE commands. Unfortunately, Atari BASIC does not have BGET and BPUT so it is limited to single byte (GET, PUT) and record (INPUT, PRINT) processing. If you're an Atari BASIC owner, you can still use block I/O (input-output) by calling the routines in ROM. The following demo shows how to do it. Even if you own BASIC XL as I do you can learn a lot about how the Atari works by trying the demo. You'll appreciate how much work BASIC XL saves you, with just these two commands.

The program below demonstrates reading and writing a block of characters to disk by poking the correct values into the IOCB (Input-Output Control Block) memory and calling the CIO (Central Input-Output Utility) to do the work. It is not meant to be a subroutine for inclusion in your own program, but only to document and demonstrate the technique. The information for this program came from Mapping the ATARI by an Chadwick. It's a book you can't be without if you do any Atari programming at all. See pages 85-88 for details about IOCBs and pages 146-147 regarding the CIO. Here's a short blow by blow description of what's going on.

LINE 50: The ML$ string contains a short machine language program which jumps to the CIO vector where the actual disk access occurs. **(Be sure to type the * and d as inverse characters.)** For the assembly language programmers in the audience the opcodes are; PLA PLA PLA TAX JMP $E456. The variable IOCB0 is the location of the first IOCB block located at memory location 832.

LINE 100: This is just to show we can use any IOCB (or channel) except #0 which is used by the editor, #6 which is used by BASIC graphics commands and #7 which is used by the printer.

LINE 110: We'll use A$ to store the data as it's read in, or the data we enter before it's written to disk.

LINE 120-130: Setup the variables to specify read or write. The variable IO is used in line 140 and 210. ICCOM is used in line 160. If we're writing data, we enter our input into A$.

LINE 160-210: Here is where we set up the IOCB for the activity we want to perform. The zero. There are 16 bytes for each IOCB so we add 16 times our IOCB number (1-5). Then we add the byte offset of the data we're poking. The two byte locations require a little math. (BASIC XL has DPOKE which pokes a number into two successive bytes automatically. What a deal!)

LINE 230: Here's where we jump off into the CIO which does the actual disk access based on the info in the IOCB. That second parameter gave me fits. It's not explicitly shown in Mapping the ATARI.

LINE 240-260: If this is a read, we convert the codes of the data read into A$ into ATASCII characters and print them on the screen. It takes longer to print them than it does to read them.

```
10 REM BLOCK DISK I/O IN ATARI BASIC
20 REM DEMO BY DAVID PELINKA 2/85
30 ? CHR$(125):? :? "BLOCK DISK I/O IN BASIC DEMO"
40 DIM F$(14),ML$(7),IO$(1)
50 ML$="hhh*LVd":IOCB0=832:REM * AND d IN ML$ are
INVERSE!
60 ? :? :? "Filename (use D:FN) ";:INPUT F$
70 ? "Read(R) or Write(W) ";:INPUT IO$
80 IF IO$<>"R" AND IO$<>"W" THEN 70
90 ? "How many bytes long ";:INPUT BYTES
100 ? "Channel # to use (1-5) ";:INPUT IOCB
110 DIM A$(BYTES)
120 IF IO$="R" THEN IO=4:ICCOM=7
130 IF IO$="W" THEN IO=8:ICCOM=11:? :? "Type
string to write to disk, ( ";BYTES;")":INPUT A$
140 OPEN #IOCB,IO,0,F$
150 REM POKE IN IOCB VALUES ==> IOCB START + IOCB#
* 16 + OFFSET
160 POKE IOCB0+IOCB*16+2,ICCOM:REM BINARY READ OR
WRITE
170 HI=INT(ADR(A$)/256):LO=ADR(A$)-HI*256
180 POKE IOCB0+IOCB*16+4,LO:POKE
IOCB0+IOCB*16+5,HI:REM BUFFER ADDRESS
190 HI=INT(BYTES/256):LO=BYTES-HI*256
200 POKE IOCB0+IOCB*16+8,LO:POKE
IOCB0+IOCB*16+9,HI:REM BUFFER LENGTH
210 POKE IOCB0+IOCB*16+10,IO:REM AUX1
220 REM JUMP TO CIOV, STACK MUST INCLUDE IOCB# *
16
230 X=USR(ADR(ML$),IOCB*16)
240 IF IO$="W" THEN ? "File written...";:GOTO 270
250 ? :? F$;" begins..."
260 FOR I=0 TO BYTES-1:CH=PEEK(ADR(A$)+I):?
CHR$(CH);:NEXT I
270 CLOSE #IOCB:CLR :GOTO 40
```

Reprinted from the
Tri-county Computer Club


```
5 REM PEEK 764 DEMO
7 POKE 764,255:REM SET VALUE
10 ? CHR$(125):POKE 710,2:POKE 755,0
20 POSITION 10,15:? "PLEASE PRESS A KEY"
25 POSITION 10,16:? "AND WATCH WHAT HAPPENS!"
30 POSITION 10,2:? " THE PEEK(764) VALUE"
35 POSITION 10,4:? " OF THE KEY IS ";PEEK764;"
40 SETCOLOR 4,PEEK764,4
45 FOR DROP=14 TO 0 STEP -1
50 SOUND 0,PEEK764+40,10,DROP
55 FOR DELAY=1 TO 5:NEXT DELAY:NEXT DROP
60 PEEK764=PEEK(764)
90 GOTO 35
```

```
5 REM KEY PRESS DEMO
10 DIM KEY$(1):POKE 710,2:? CHR$(125)
20 POSITION 12,15:? "PLEASE PRESS A KEY"
25 POSITION 12,16:? "AND THEN RETURN";:INPUT KEY$
26 POSITION 27,16:? "     "
30 POSITION 10,2:? KEY$;" IS THE KEY YOU PRESSED"
40 POSITION 10,4:? "THE CHR$ VALUE IS ";ASC(KEY$);"  "
55 SETCOLOR 4,ASC(KEY$),4
60 FOR DROP=14 TO 0 STEP -1
70 SOUND 0,ASC(KEY$),10,DROP
75 FOR DELAY=1 TO 10:NEXT DELAY:NEXT DROP
100 GOTO 20
```

MILWAUKEE AREA ATARI USER'S GROUP AND NEWSLETTER INFORMATION

MILATARI OFFICERS

President	David Frazer	542-7242
Vice President	Carl Mielcarek	355-3539
Secretary	Jim Mangione	764-6855
Treasurer	Steve Tupper	462-8178

MILATARI VOLUNTEERS

MILATARI West	Dennis Bogie	968-2361
Education SIG	Joe Sanders	447-1660
ATR8000 & CP/M	Joe Kasper	782-9041
Adv. Lang. SIG	Erik Hanson	252-3146
Database	Ron Friedel	354-1717
BBS Sysop	Richard Dankert	781-2338
Membership	Dave Bogie	968-2361
Newsletter	Roy Duvall	363-8231
Public Domain Libraries		
Cassette	Lee Musial	466-9160
Disk	Dennis Wilson	476-7767
	Bill Lawrence	968-3082
Copyright Library		
Cassette/Disk	Gary Haberman	228-8845
Publications	Bill Feest	321-4314
	Milatari BBS	
	414-781-5710	

NEWSLETTER INFORMATION

This newsletter is written and printed by members of the Milwaukee Area ATARI User's Group (MILATARI). Opinions expressed in this publication are those of the individual authors and do not necessarily represent or reflect the opinions of the Milwaukee Area Atari User's Group, its officers, its members or its advertisers except where noted. Articles reprinted from bulletin boards and other newsletters are presented as an information exchange, no warranty is made to their accuracy or factuality.

Your contributions of articles are always welcome. You may submit your article on ATARI compatible cassette or diskette, on typewritten form or you can arrange with the editor to upload your file via modem. You can send Graphics eight or seven plus screens stored on disk in Micropainter or Micro Illustrator formats.

Other computer user groups may obtain copies of this newsletter on an exchange basis.

Milwaukee Area Atari User's Group

MILATARI is an independent, user education group which is not affiliated with ATARI INC. The newsletter is the official publication of MILATARI and is intended for the education of its members as well as for the dissemination of information concerning ATARI computer products.

MILATARI membership is open to individuals and families who are interested in using and programming ATARI computers. The membership includes a subscription to this newsletter and access to the club libraries. The annual membership fee is $15 for individuals or $20 for a family.

Vendors wishing to display and/or sell items at MILATARI meetings must make prior arrangements with the club vice president. Rates are $10 per meeting or $90 per year payable in advance.

MILATARI ADVERTISING RATES

This newsletter will accept camera ready advertising copy from anyone supplying goods and services of interest to our membership.

Current paid members of MILATARI may place classified ads in the newsletter at no charge.

Advertising Rates

Full page	$37.50
Half page	$20.00
Quarter page	$12.50
Business card	$2.00

MILWAUKEE AREA ATARI USERS CLUB

U.S. POSTAGE PAID
BULK RATE
PERMIT NO. 201
WAUKESHA, WI 53187

MILWAUKEE AREA ATARI USERS GROUP
Post Office Box 19858
West Allis, Wisconsin 53219-0858

==========

Address Correction Requested
Forwarding Postage Guaranteed

==========